13

Rendering Data with ASP.NET Web Controls

13.0. Introduction

ASP.NET provides several Web controls that make displaying data on a Web page easier than ever before. This chapter shows you how to take advantage of a process called data-binding to easily display data in a variety of formats using very little code. This chapter covers many of the most commonly used features of the Repeater, DataList, and DataGrid, including some fairly advanced DataGrid features. Using these data-bound Web controls, it is very easy to write data-driven Web Forms by just dragging and dropping a few controls onto a form and writing a few lines of code.

13.1. Rendering Data Directly on a Web Form

You want to display a piece of data on a Web Form using data-binding.

Technique

You can use the `<%#%>` syntax to easily bind data to a control. Simply create a variable, assign it a value, and call `Page.DataBind` to bind it to the page. The ASPX page is as follows:

```
<html>
  <body>
   <form id="Recipe1401vb" method="post" runat="server">
      <asp:Label ID="MyLabel" Runat="server">Hello <%#FirstName%>!</asp:Label>
   </form>
  </body>
</html>
```

In `<script runat="server" />` block or codebehind:

```
Protected FirstName as string

Sub Page_Load(sender As Object, e As EventArgs)
   FirstName="Jeremy"
   Page.DataBind()
End Sub
```

Comments

If you are using codebehind, it is important to declare the variable with `Protected` access level so the page can access it. It is also important to realize that calling `Page.DataBind()` will result in all controls on the page being data-bound, because any time a control container calls its `DataBind()` method, it recursively calls the `DataBind()` method of all of its child controls.

See Also

Section 13.2, "Data-binding to a DropDownList"

Section 13.3, "Data-binding to a Repeater"

Section 13.4, "Data-binding to a DataList"

Section 13.7, "Data-binding to a DataGrid"

13.2. Data-binding to a DropDownList

You want to create a DropDownList that is populated from a database.

Technique

To data-bind a DropDownList to a data container such as a dataset or datareader, you must set three properties and call one method. The following example demonstrates this.

In `<script runat="server" />` block or codebehind:

```
Sub Page_Load(sender As Object, e As EventArgs)
    'object vars
    Dim sqlConnection As SqlConnection
    Dim sqlDataAdapter As SqlDataAdapter
    Dim sqlCommand As SqlCommand
    Dim dataSet As DataSet
  Dim dataTable As DataTable

    Try
        sqlConnection = New SqlConnection("Integrated Security=yes;
```

```
➥Initial Catalog=Northwind;Data Source=(local)")

            'pass the stored proc name and SqlConnection
            sqlCommand = New SqlCommand("Select * From Customers", _
                sqlConnection)

            'instantiate SqlAdapter and DataSet
            sqlDataAdapter = New SqlDataAdapter(sqlCommand)
            dataSet = New DataSet()

            'populate the DataSet
            sqlDataAdapter.Fill(dataSet, "Customers")

         'apply sort to the DefaultView to sort by CompanyName
            dataSet.Tables(0).DefaultView.Sort = "CompanyName"

            DropDownList1.DataSource = dataSet.Tables("Customers").DefaultView
         DropDownList1.DataTextField = "CompanyName" ' what to display
         DropDownList1.DataValueField = "CustomerID" ' what to set as value
         DropDownList1.DataBind()

        Catch exception As Exception
            errorMsgLabel.Text = exception.ToString()

        End Try
    End Sub
```

Comments

As a general rule, you can perform this kind of data-binding only when the form first loads, as opposed to on every postback. To do this, simply place the data access and data-binding code within an `If` statement so that they are only performed when `Page.IsPostback` is `false` (for example, on the first load of the page). You can manipulate the `Items` collection after calling `DataBind()` if you need to set a particular item as `Selected` (use `Items.FindByText` or `Items.FindByValue` to find the item you want) or insert a default entry for the first item (use `Items.Insert()` with an index of `0` for the first item).

See Also

ASP.NET DropDownList I—`http://aspalliance.com/stevesmith/articles/dotnetlistbox1.asp`

ASP.NET DropDownList II—`http://aspalliance.com/stevesmith/articles/dotnetlistbox2.asp`

Dynamically Set Text and Value of DropDownList—`http://aspalliance.com/aldotnet/examples/dynamicdatasource.aspx`

13.3. Data-binding to a Repeater

You want to use a repeater to output the results of a data query.

Technique

The Repeater control is the simplest of three templated data-bound controls provided with ASP.NET (the others being the DataList and DataGrid). It supports templates for Header, Item, AlternatingItem, Separator, and Footer, which can each contain static and dynamic (data-bound) content. The following example demonstrates how to set up a Repeater's templates and how to data-bind the Repeater to a dataset.

The ASPX page is as follows:

```
<asp:Repeater id="Repeater1" runat="server">
   <HeaderTemplate>Customers:<br/><ul></HeaderTemplate>
   <ItemTemplate>
      <li><%#DataBinder.Eval(Container.DataItem, "CompanyName")%>,
         <%#DataBinder.Eval(Container.DataItem, "ContactName")%></li>
   </ItemTemplate>
   <AlternatingItemTemplate>
      <li><font color="red">
<%#DataBinder.Eval(Container.DataItem, "CompanyName")%>,
<%#DataBinder.Eval(Container.DataItem, "ContactName")%></font></li>
   </AlternatingItemTemplate>
   <FooterTemplate><hr/>Data Retrieved at:
<%# System.DateTime.Now.ToString() %></FooterTemplate>
</asp:Repeater>
             <br>
             <br>
             <asp:Label id="errorMsgLabel" runat="server"
Width="327px" Height="111px"></asp:Label>
```

In `<script runat="server" />` block or codebehind:

```
Sub BindRepeater()
    'object vars
    Dim sqlConnection As SqlConnection
    Dim sqlDataAdapter As SqlDataAdapter
    Dim sqlCommand As SqlCommand
    Dim dataSet As DataSet
   Dim dataTable As DataTable

    Try
        sqlConnection = New SqlConnection("Integrated Security=yes;
➥Initial Catalog=Northwind;Data Source=(local)")

        'pass the stored proc name and SqlConnection
```

```
        sqlCommand = New SqlCommand("Select * From Customers", sqlConnection)

        'instantiate SqlAdapter and DataSet
        sqlDataAdapter = New SqlDataAdapter(sqlCommand)
        dataSet = New DataSet()

        'populate the DataSet
        sqlDataAdapter.Fill(dataSet, "Customers")

      'apply sort to the DefaultView to sort by CompanyName
        dataSet.Tables(0).DefaultView.Sort = "CompanyName"

        Repeater1.DataSource = dataSet.Tables("Customers").DefaultView
      Repeater1.DataBind()

    Catch exception As Exception
        errorMsgLabel.Text = exception.ToString()

    End Try
End Sub
```

Comments

`BindRepeater()` is called by `Page_Load` only when it is first loaded (not after a postback). It will automatically retain its state between postbacks using ViewState, thus avoiding additional requests to the database.

> **See Also**
> Repeater Class—`http://msdn.microsoft.com/library/en-us/cpref/html/frlrfsystemwebuiwebcontrolsrepeaterclasstopic.asp`

13.4. Data-binding to a DataList

You want to use a DataList control to output the results of a data query.

Technique

Create a DataList and set the layout and visual formatting the way you want. Inside the DataList, add an `ItemTemplate` tag. Inside this tag, you can set static HTML code to display along with pieces of data. To insert a piece of data, use the following syntax:

`<%#DataBinder.Eval(Container.DataItem, "FieldName")%>.`

After you have the DataList set up, populate a DataTable (or any class that implements the `System.Collections.IEnumerable` interface). Then set the `DataSource` property of the DataList to point to your DataTable, and call the `DataBind()` method of the DataList.

The ASPX page is as follows:

```
<%@Import namespace="System.Data" %>
<%@Import namespace="System.Data.SqlClient" %>
<html>
  <body>
    <form id="Recipe1404vb" method="post" runat="server">
     <asp:DataList id="MyDataList" Runat=server
RepeatDirection=Horizontal RepeatColumns=2 ItemStyle-BorderWidth=1>
       <ItemTemplate>
         Name: <%#DataBinder.Eval(Container.DataItem, "CategoryName")%><br>
         Description: <%#DataBinder.Eval(Container.DataItem, "Description")%>
       </ItemTemplate>
     </asp:DataList>
     </form>
  </body>
</html>
```

In `<script runat="server" />` block or codebehind:

```
Sub Page_Load(sender As Object, e As EventArgs)
   Dim myTable As New DataTable()
   Dim myConn As _
   New SqlConnection("Server=localhost;Database=Northwind;UID=sa;PWD=")
   Dim myAdapter As New SqlDataAdapter(
➥"Select CategoryName, Description FROM Categories", myConn)
   myAdapter.Fill(myTable)
   MyDataList.DataSource=myTable
   MyDataList.DataBind()
End Sub
```

Comments

This code uses the Northwind database that comes with the default installation of Microsoft SQL Server.

See Also

Section 10.1, "Connecting to SQL Server"

Section 11.1, "Executing a Stored Procedure and Returning the Results in a Datareader"

Section 13.2, "Data-binding to a DropDownList"

Section 13.3, "Data-binding to a Repeater"

Section 13.7, "Data-binding to a DataGrid"

13.5. Implementing Sorting in a DataList

You want to implement sorting in your DataList.

Technique

Create a DataList and set the data items to display in the `ItemTemplate` as outlined in example 13.4. Add a header template with link buttons `SortByName` and `SortByDescription`. `OnClick` events should be wired to the `SortByName_OnClick` and `SortByDescription_OnClick` methods.

Create the `PullCategories()` method to query the categories table and populate `myView` and the `PopulateCategories()` method to bind `myView` to `MyDataList`. In the `Page_Load` event, determine whether this is the first load of the page and, if so, populate the DataList with the default sort. In the `SortByName_Click` event, call `PullCategories()`, set the `Sort` property of `myView` to `CategoryName` and call the `PopulateCategories()` method. Do the same for `SortByDescription_Click`, setting the sort to `Description`.

The ASPX page is as follows:

```
<%@Import namespace="System.Data.SqlClient" %>
<%@Import namespace="System.Data" %>

<HTML>
   <body>
      <form id="Recipe1405vb" method="post" runat="server">
        <asp:DataList id="MyDataList" Runat="server" ItemStyle-BorderWidth="1">
            <HeaderTemplate>
               Sort By
            <asp:LinkButton ID="SortByName" Runat="server"
 OnClick="SortByName_Click">Name</asp:LinkButton> /
               <asp:LinkButton ID="SortByDescription" Runat="server"
OnClick="SortByDescription_Click">Description</asp:LinkButton>
            </HeaderTemplate>
            <ItemTemplate>
               <%#DataBinder.Eval(Container.DataItem, "CategoryName")%>
               <br>
               <%#DataBinder.Eval(Container.DataItem, "Description")%>
            </ItemTemplate>
         </asp:DataList>
      </form>
   </body>
</HTML>
```

In `<script runat="server" />` block or codebehind:

```
private myView AS DataView
```

```
Private Sub Page_Load(sender As Object, e As EventArgs)
   If Not IsPostBack Then
      PullCategories()
      myView.Sort="CategoryName"
      PopulateCategories()
   End If
End Sub

Protected Sub SortByName_Click(sender As Object, e As EventArgs)
   PullCategories()
   myView.Sort="CategoryName"
   PopulateCategories()
End Sub

Protected Sub SortByDescription_Click(sender As Object, e As EventArgs)
   PullCategories()
   myView.Sort="Description"
   PopulateCategories()
End Sub

Private Sub PullCategories()
   Dim myTable As New DataTable()
   Dim myConn As New SqlConnection("Server=localhost;
➥Database=Northwind;UID=sa;PWD=")
   Dim myAdapter As New SqlDataAdapter(
➥"Select CategoryName, Description FROM Categories", myConn)
   myAdapter.Fill(myTable)
   myView=New DataView(myTable)
End Sub

Private Sub PopulateCategories()
   MyDataList.DataSource=myView
   MyDataList.DataBind()
End Sub
```

Comments

This code uses the Northwind database that comes with the default installation of Microsoft SQL Server.

See Also

Section 13.4, "Data-binding to a DataList"

Section 13.6, "Implementing Paging in a DataList"

Section 13.11, "Enabling Sorting in a DataGrid"

Section 13.12, "Enabling Bi-Directional Sorting in a DataGrid"

13.6. Implementing Paging in a DataList

You want to implement paging in your DataList.

Technique

By default, the DataList control does not support paging—only the DataGrid has that functionality built in. However, you can implement your own custom paging solution with the DataList by following these steps:

1. Add paging controls (Next/Prev at a minimum, but perhaps also First, Last, Next 5, Prev 5, and so on).
2. Set and maintain a `PageSize` variable.
3. Modify DataSource prior to data-binding to ensure the records for the appropriate page are the only ones bound.

For Step 1, consider the following pager construct within the `FooterTemplate` of the following DataList:

```
<asp:DataList id="DataList1" runat="server"
RepeatDirection="Horizontal" RepeatColumns="3">
   <HeaderTemplate>
      Customers
   </HeaderTemplate>
   <FooterTemplate>
      <!— Pager Construct —>
      <table width="100%" align="right">
         <tr>
            <td width="76%" align="left">
               <asp:Label ID="StatusLabel" Runat="server"
Font-Name="verdana" Font-Size="10pt" />
            </td>
            <td width="6%">
               <a href="datalistpaging.aspx#this"
ID="hrefFirst" onserverclick="ShowFirst" runat="server">
                  <<<</a>
            </td>
            <td width="6%">
               <a href="datalistpaging.aspx#this"
ID="hrefPrevious" onserverclick="ShowPrevious" runat="server">
                  <<</a>
            </td>
            <td width="6%">
               <a href="datalistpaging.aspx#this"
ID="hrefNext" onserverclick="ShowNext" runat="server">
                  >></a>
```

```
            </td>
            <td width="6%">
               <a href="datalistpaging.aspx#this"
ID="hrefLast" onserverclick="ShowLast" runat="server">
                  >>></a>
            </td>
         </tr>
      </table>
   </FooterTemplate>
   <ItemTemplate>
      <table border="1" cellpadding="0" cellspacing="0">
         <tr>
            <td>
               Company:
               <%#DataBinder.Eval(Container.DataItem, "CompanyName")%>
               <br />
               Contact:
               <%#DataBinder.Eval(Container.DataItem, "ContactName")%>
            </td>
         </tr>
      </table>
   </ItemTemplate>
   <AlternatingItemTemplate>
      <table border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
         <tr>
            <td>
               Company:
               <%#DataBinder.Eval(Container.DataItem, "CompanyName")%>
               <br />
               Contact:
               <%#DataBinder.Eval(Container.DataItem, "ContactName")%>
            </td>
         </tr>
      </table>
   </AlternatingItemTemplate>
</asp:DataList>
```

The pager code includes a label to display the current position in the records, a first page link, a last page link, and next and previous links. Each of the links is wired to an event through the use of the `onserverclick` attribute. In the code, the page retrieves the data from the database as a DataTable and stores it in ViewState so that subsequent requests do not require further data access. Then each event handler adjusts the `currentPage` variable that is also maintained in ViewState. Using the current page index and the page size, the `ShowPage()` method pulls out just the records that should be displayed on that page and places them into a new DataTable, which is then bound to the DataList.

In `<script runat="server" />` block or codebehind:

```
Dim dataTable As DataTable = Nothing
Dim pageSize As Integer = 5
Dim currentPage As Integer

Sub Page_Load(sender As Object, e As EventArgs)
   If Not IsPostBack Then
      ShowFirst(Me, System.EventArgs.Empty)
   End If
End Sub

Sub GetData()
   If ViewState("Data") Is Nothing Then
      'object vars
      Dim sqlConnection As SqlConnection
      Dim sqlDataAdapter As SqlDataAdapter
      Dim sqlCommand As SqlCommand

      Try
         Trace.Write("GetData","Getting data from database.")
         sqlConnection = New SqlConnection("Integrated Security=yes;
➥Initial Catalog=Northwind;Data Source=(local)")

         'pass the stored proc name and SqlConnection
         sqlCommand = New SqlCommand(
➥"Select * From Customers Order By CompanyName", sqlConnection)

         'instantiate SqlAdapter and DataTable
         sqlDataAdapter = New SqlDataAdapter(sqlCommand)
         dataTable = New DataTable()

         'populate the DataTable
         sqlDataAdapter.Fill(dataTable)

           ViewState("Data") = dataTable

      Catch exception As Exception
         ErrorLabel.Text = exception.ToString()
      End Try
   Else
      Trace.Write("GetData","Getting data from ViewState.")
      dataTable = CType(ViewState("Data"), DataTable)
    End If
End Sub

Sub ShowFirst(sender As Object, e As EventArgs)
   currentPage = 1
```

```
   ShowPage()
End Sub

Sub ShowLast(sender As Object, e As EventArgs)
   GetData()
   currentPage = CType(System.Math.Ceiling(CType(
➥dataTable.Rows.Count,Double) / pageSize), Integer)
   ShowPage()
End Sub

Sub ShowNext(sender As Object, e As EventArgs)
   GetData()
   currentPage = CType(ViewState("CurrentPage"), Integer)
   If  currentPage <= CType(System.Math.Ceiling(CType(
➥dataTable.Rows.Count,Double) / pageSize), Integer) Then
      currentPage += 1
   End If
   ShowPage()
End Sub

Sub ShowPrevious(sender As Object, e As EventArgs)
   currentPage = CType(ViewState("CurrentPage"), Integer)
   If currentPage > 1 Then
      currentPage -= 1
   End If
   ShowPage()
End Sub

Sub ShowPage()
   Dim I As Integer
   If dataTable Is Nothing Then
      GetData()
   End If
   If dataTable Is Nothing Then
      Throw New ApplicationException("Data failed to load.")
   End If

   Dim dt2 As DataTable = dataTable.Clone()
'Copy the structure of the data to a new container

   For I = ((currentPage-1)*pageSize) To (currentPage*pageSize) -1
      If I >= dataTable.Rows.Count Then Exit For
      dt2.ImportRow(dataTable.Rows(I))
   Next

   DataList1.DataSource = dt2
   DataList1.DataBind()
```

```
    'Display status line
    CType(DataList1.Controls(DataList1.Controls.Count-1).
➥FindControl("StatusLabel"), Label).Text = _
        "Total Records: " & _
dataTable.Rows.Count & ".  Page " & currentPage & " of " & _
        CType(System.Math.Ceiling(CType(
➥dataTable.Rows.Count,Double) / pageSize), Integer) & "."
    ' Store current page
    ViewState("CurrentPage") = currentPage
End Sub
```

Comments

This example uses the same kind of paging as the default behavior of the DataGrid. That is, the records are retrieved once from the database and from then on they are passed back and forth with each request in the page's ViewState. This design choice has advantages and disadvantages, which you must consider before you choose to implement paging in this fashion in your solutions.

The advantage is that it greatly reduces the number of requests that must be made to the database server, which is important because the database is typically one of the most difficult pieces of an application to scale upward if it becomes the bottleneck.

The disadvantage is that all of the data is being passed to and from the client with each request. Obviously for extremely large result sets, this is not ideal. Even for relatively small result sets, this can be a problem when the client is not using a high-speed network connection.

An alternative approach is to write a query or stored procedure that takes a page index and page size and returns only those records needed for the specified page. This results in more database resources being consumed, but fewer network and Web server resources and less dependence on the client's connection speed. For more on this approach, see section 13.10, "Enabling Custom Paging in a DataGrid."

See Also

Section 13.10, "Enabling Custom Paging in a DataGrid"

13.7. Data-binding to a DataGrid

You want to use a DataGrid control to display the results of a data query.

Technique

Use `System.Data.SqlClient.SqlDataAdapter` to retrieve data from a SQL Server database as a `System.Data.DataSet` and bind the results to `System.Web.UI.WebControls.DataGrid` in a Web page.

Import the following namespaces into your page:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
```

In `<script runat="server" />` block:

```
Sub Page_Load(sender As Object, e As EventArgs)
  Dim PubsDataSet As DataSet
  Dim PubsConnection As SqlConnection
  Dim PubsAdapter As SqlDataAdapter
  PubsConnection = New _
SqlConnection("server=localhost;database=pubs;Trusted_Connection=yes")
  PubsAdapter = New SqlDataAdapter(
➥"SELECT * FROM Publishers", PubsConnection)
  PubsDataSet = New DataSet()
Try
    PubsConnection.Open()
    PubsAdapter.Fill(PubsDataSet, "Publishers")
    PublisherDataGrid.DataSource = PubsDataSet
    PublisherDataGrid.DataMember = "Publishers"
    PublisherDataGrid.DataBind()
  Catch PubsException As SqlException
    PubsExceptionLabel.Text = PubsException.Message
  Finally
    PubsConnection.Close()
  End Try
End Sub
```

The ASPX page is as follows:

```
<html>
  <body>
    <asp:DataGrid id="PublisherDataGrid" EnableViewState="false"
         runat="server" />
    <asp:Label id="PubsExceptionLabel" EnableViewState="false"
         runat="server" />
  </body>
</html>
```

Comments

The SQL `SELECT` statement determines which rows and columns are displayed in the DataGrid. You can change the appearance (border, color, font) of the DataGrid by setting the DataGrid's style properties. It's best to store the connection string information in the application settings of the web.config file. In addition to `System.Data.SqlClient.SqlDataAdapter`, `System.Data.Oledb.OledbDataAdapter` allows you to display data from Access, Oracle and other OleDb databases.

See Also

Section 6.2, "Creating Custom Application Settings in the web.config File"

Section 10.1, "Connecting to SQL Server"

Section 11.3, "Filtering the Contents of a Dataset"

Populating a Dataset from a DataAdapter—`http://msdn.microsoft.com/library/en-us/cpguide/html/cpconpopulatingdatasetfromdataadapter.asp`

13.8. Using Different Column Types in a DataGrid

You want to display data, buttons, or custom formatted output within a DataGrid using its various column types.

Technique

The DataGrid has five types of columns—Bound, Button, EditCommand, HyperLink, and Template. Each one of these provides a different functionality, as you will see in this example.

The `BoundColumn` directly binds a column from the datasource to the DataGrid. In the .ASPX file:

```
<asp:DataGrid ...>
<Columns>
<asp:BoundColumn
   HeaderText="Phone Number"
   DataField="PhoneNo"
   ReadOnly="False"
   DataFormatString="{0:N}" />
</Columns>
</asp:DataGrid>
```

The `HeaderText` specifies the text for the column header, `DataField` specifies the field in the datasource to bind to the column, `ReadOnly` specifies whether the column can be edited in edit mode, and `DataFormatString` specifies how to format the column (in the example, as a number).

The `ButtonColumn` displays a command button in each row of the DataGrid. In the .ASPX file:

```
<asp:DataGrid ...>
<Columns>
<asp:ButtonColumn
   HeaderText="Remove Contact"
   ButtonType="LinkButton"
   Text="Remove"
```

```
   CommandName="RemoveContact" />
</Columns>
</asp:DataGrid>
```

The `ButtonType` specifies the type of command button to use; this can be set to `PushButton` or `LinkButton`. You can specify `DataTextField` and `DataTextFormatString` instead of `Text` in the tag. This would set the text value from a field in the datasource (`DataTextField`), and format it according to the `DataTextFormatString` specified.

The `EditCommandColumn` displays a column with command buttons for editing the data in each row. In the .ASPX file:

```
<asp:DataGrid ...>
<Columns>
<asp:EditCommandColumn
   ButtonType="LinkButton"
   UpdateText="Save"
   CancelText="Cancel"
   EditText="Edit" />
</Columns>
</asp:DataGrid>
```

`HyperLinkColumn` displays a hyperlink in each row of the DataGrid. In the .ASPX file:

```
<asp:DataGrid ...>
<Columns>
<asp:HyperLinkColumn
   Text="View Contact Details"
   DataNavigateUrlField="UserID"
   DataNavigateUrlFormatString="userdetails.aspx?userid={0}"
   Target="_new" />
</Columns>
</asp:DataGrid>
```

The `DataNavigateUrlField` and `DataNavifateUrlFormatString` properties determine the field and formatting string to use when setting the link. You can also use `DataTextField` and `DataTextFormatString` instead of the `Text` field (as you can with `ButtonColumn`). Finally, you can use the `NavigateUrl` property instead of the `DataNavigateUrl` and `DataNavigateUrlFormatString` properties to set a static URL.

The `TemplateColumn` provides the most functionality, because it allows you to specify a customized column layout. In the .ASPX page:

```
<asp:DataGrid ...>
<Columns>
<asp:TemplateColumn>
<HeaderTemplate>
Name
</HeaderTemplate>
<ItemTemplate>
```

```
<asp:Label runat="server"
   Text='<%# Container.DataItem("FirstName") & " " & _
Container.DataItem("LastName") %>' />
</ItemTemplate>
<EditItemTemplate>
First Name : <asp:TextBox runat="server"
      Text='<%# Container.DataItem("FirstName") %>' />
<br>
Last Name : <asp:TextBox runat="server"
      Text='<%# Container.DataItem("LastName") %>' />
</EditItemTemplate>
<FooterTemplate>
<asp:HyperLink runat="server"
   Text="Go Home"
   NavigateUrl="default.aspx"  />
</FooterTemplate>
</asp:TemplateColumn>
</Columns>
</asp:DataGrid>
```

Comments

The different types of DataGrid columns provide a great deal of functionality when developing a DataGrid, as you can see in these examples. Each of the column types inherit from the `DataGridColumn` class and therefore provide some basic functionality within all of them, such as visibility and styles.

See Also

The DataGridColumn Class—`http://msdn.microsoft.com/library/en-us/cpref/html/frlrfsystemwebuiwebcontrolsdatagridcolumnclasstopic.asp`

The DataGrid Class—`http://msdn.microsoft.com/library/en-us/cpref/html/frlrfsystemwebuiwebcontrolsdatagridclasstopic.asp`

Section 15.12, "Formatting Strings"

13.9. Enabling Default Paging in a DataGrid

You want to enable paging in your DataGrid.

Technique

Use the `System.Web.UI.WebControls.DataGrid.OnPageIndexChanged` event to set the `System.Web.UI.WebControls.DataGrid.CurrentPageIndex` property equal to `System.Web.UI.WebControls.DataGridSortCommandEventArgs.NewPageIndex`.

Import the following namespaces into your page:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
```

In `<script runat="server" />` block or codebehind:

```
Sub Page_Load(Sender As Object, E As EventArgs)
  If Not Page.IsPostBack Then
    BindGrid()
  End If
End Sub

Sub BindGrid()
  Dim PubsDataSet As DataSet
  Dim PubsConnection As SqlConnection
  Dim PubsAdapter As SqlDataAdapter
  PubsConnection = New _
SqlConnection("server=localhost;database=pubs;Trusted_Connection=yes")
  PubsAdapter = New SqlDataAdapter(
➥"select * from Publishers", PubsConnection)
  PubsDataSet = new DataSet()
Try
    PubsConnection.Open()
    PubsAdapter.Fill(PubsDataSet, "Publishers")
    PublisherDataGrid.DataSource = PubsDataSet
    PublisherDataGrid.DataMember = "Publishers"
    PublisherDataGrid.DataBind()
  Catch PubsException As SqlException
    PubsExceptionLabel.Text = PubsException.Message
  Finally
    PubsConnection.Close()
  End Try
End Sub

Sub PublisherDataGrid_Page(Sender As Object, _
E As DataGridPageChangedEventArgs)
  PublisherDataGrid.CurrentPageIndex = e.NewPageIndex
  BindGrid()
End Sub
```

The ASPX page:

```
<html>
  <body>
    <form runat="server">
      <asp:DataGrid id="PublisherDataGrid" runat="server"
        OnPageIndexChanged="PublisherDataGrid_Page"
```

```
            PageSize="3"
            AllowPaging="True"/>
          <asp:Label id="PubsExceptionLabel" EnableViewState="false"
runat="server" />
        </form>
      </body>
    </html>
```

Comments

The `System.Web.UI.WebControls.DataGrid.PageSize` property has a default of 10. The `System.Web.UI.WebControls.DataGrid.PagerStyle-Mode` property default is `NextPrev`. For numeric page links, you should use `NumericPages`.

> **See Also**
>
> DataGrid Paging in a Web Form—`http://msdn.microsoft.com/library/en-us/dnvssamp/html/vbcs_PagingThroughQueryResults.asp`
>
> Specifying Paging Behavior in a DataGrid Web Server Control—`http://msdn.microsoft.com/library/en-us/vbcon/html/vbtskspecifyingpagingbehaviorindatagridwebcontrol.asp`

13.10. Enabling Custom Paging in a DataGrid

You want to configure a custom paging solution for a DataGrid.

Technique

Using custom paging in the DataGrid control, you can retrieve and display only those records needed for the current request. In order to implement the custom paging feature of DataGrid, you have to adjust four properties first—`PageSize`, `AllowPaging`, `AllowCustomPaging`, and `VirtualItemCount`.

The `PageSize` property determines the number of records displayed per page on the DataGrid. The `AllowPaging` property enables pagination of the DataGrid. `AllowCustomPaging` determines whether the default paging behavior should be used. In this example, you are replacing the default paging behavior with your own. The `VirtualItemCount` determines the total number of items in the DataGrid control when custom paging is used.

The ASPX page:

```
<asp:datagrid id="myDataGrid" runat="server"
  width="100%"
  OnPageIndexChanged="myDataGrid_Page"
```

```
      PageSize="6"
      AllowCustomPaging="True"
      AllowPaging="True">

                <PagerStyle mode="NumericPages"/>

            </asp:datagrid>
```

You have to retrieve the total number of records in the database table first—that number is used for the `VirtualItemCount`. After that, the `OnPageIndexChanged` event handler will be executed if the page number link was clicked. In this event handler, you have to change the current page index of the DataGrid as usual, but you also have to calculate the starting and ending index for the record. This is calculated by multiplying the index of the selected page by the page size.

In `<script runat="server" />` block or codebehind:

```
Sub myDataGrid_Page(Sender As Object, _
e As DataGridPageChangedEventArgs)

    myDataGrid.CurrentPageIndex = e.NewPageIndex
    BindGrid(e.NewPageIndex * myDataGrid.PageSize)

End Sub
```

In the BindGrid procedure, you have to calculate the starting index of the record by multiplying the index of the selected page by the page size. The ending index of the records is the sum of the starting index and the page size of DataGrid. Using these starting and ending indexes, you can retrieve the records within this range.

```
Sub BindGrid(StartingIndex As Integer)

   Dim startingID As Integer = StartingIndex
   Dim endingID As Integer = startingID + myDataGrid.PageSize

   Dim ConnectionString As String = _
"server=(local);database=Northwind;trusted_connection=true"
   Dim CommandText As String = "select ProductID, " & _
                        "ProductName, " & _
                        "UnitPrice, " & _
                        "UnitsInStock " & _
                        "from Products " & _
                        "where ProductID > @StartingID and " & _
                        "ProductID <= @EndingID"

   Dim objConnection As New SqlConnection(ConnectionString)
   Dim objCommand As New SqlDataAdapter(CommandText, objConnection)
```

```
   Try

      objCommand.SelectCommand.Parameters.Add("@StartingID", startingID)
      objCommand.SelectCommand.Parameters.Add("@EndingID", endingID)

      Dim objDataSet As New DataSet()
      objCommand.Fill(objDataSet)

      myDataGrid.DataSource = objDataSet.Tables(0).DefaultView
      myDataGrid.VirtualItemCount = GetVirtualItemCount()
      myDataGrid.DataBind()

   Catch SqlEx As SqlException
      MessageLabel.Text = SqlEx.Message

   Catch Ex As Exception
      MessageLabel.Text = Ex.Message

   Finally
      objConnection.Close()

   End Try

End Sub
   Function GetVirtualItemCount As Integer

      If ViewState("GetVirtualItemCount") Is Nothing Then
         Dim ConnectionString As String = _
"server=(local);database=Northwind;trusted_connection=true"
         Dim CommandText As String = "select count(*) from Products"
         Dim objConnection As New SqlConnection(ConnectionString)
         Dim objCommand As New SqlCommand(CommandText, objConnection)
         Try
            objConnection.Open()
            ViewState("GetVirtualItemCount") = _
CType(objCommand.ExecuteScalar(), Integer)
         Catch SqlEx As SqlException
            MessageLabel.Text = SqlEx.Message
         Catch Ex As Exception
            MessageLabel.Text = Ex.Message
         Finally
            objConnection.Close()
         End Try
      End If
      Return CInt(ViewState("GetVirtualItemCount"))
   End Function
```

Comments

Even though there is built-in pagination capability in the DataGrid control, custom paging gives you better performance. When you enable the built-in paging, all the records must be retrieved from the datasource in your first page load or page index changed. Then all of these records must be passed to and from the browser in ViewState on each subsequent page request.

Using custom paging, only the rows needed for each page are retrieved. This will make a huge difference in a paged DataGrid with 1,000,000 records.

One important assumption made here is that the database table has a unique column that can be used by the data retrieval procedure for its starting and ending points.

See Also

Section 13.9, "Enabling Default Paging in a DataGrid"

13.11. Enabling Sorting in a DataGrid

You want to enable sorting in certain columns of your DataGrid.

Technique

Use the `System.Web.UI.WebControls.DataGrid.SortCommand` event to set the `System.Data.DataView.Sort` property equal to the `System.Web.UI.WebControls.DataGridColumn.SortExpression` property.

Import the following namespaces into your page:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
```

In `<script runat="server" />` block:

```
Sub Page_Load(Sender As Object, E As EventArgs)
If Not Page.IsPostBack Then
    BindGrid("pub_id")
End If
End Sub

Sub BindGrid(SortColumn As String)
Dim PubsDataSet As DataSet
Dim PubsConnection As SqlConnection
Dim PubsAdapter As SqlDataAdapter
Dim PublishersDataView As DataView
PubsConnection = New SqlConnection("server=localhost
➥;database=pubs;Trusted_Connection=yes")
PubsAdapter = New SqlDataAdapter("select * from Publishers", PubsConnection)
```

```
PubsDataSet = new DataSet()
Try
    PubsConnection.Open()
    PubsAdapter.Fill(PubsDataSet, "Publishers")
    PublishersDataView = PubsDataSet.Tables(0).DefaultView
    PublishersDataView.Sort = SortColumn
    PublisherDataGrid.DataSource = PublishersDataView
    PublisherDataGrid.DataBind()
Catch PubsException As SqlException
    PubsExceptionLabel.Text = PubsException.Message
Finally
    PubsConnection.Close()
End Try
End Sub

Sub PublisherDataGrid_Sort(Sender As Object, E As DataGridSortCommandEventArgs)
   BindGrid(E.SortExpression)
End Sub
```

The ASPX page:

```
      <html>
        <body>
          <form runat="server">
            <asp:DataGrid id="PublisherDataGrid" runat="server"
              OnSortCommand="PublisherDataGrid_Sort"
              AllowSorting="True"/>
            <asp:Label id="PubsExceptionLabel"
EnableViewState="false" runat="server" />
          </form>
        </body>
      </html>
```

Comments

You can set the `System.Web.UI.WebControls.DataGridColumn.SortExpression` property for each column that you want to sort. When using `System.Web.UI.WebControls.DataGrid.AutoGenerateColumns`, the default used for `System.Web.UI.WebControls.DataGridColumn.SortExpression` is the `System.Web.UI.WebControls.DataGridColumn.DataColumn.DataField` property.

See Also

Adding Sorting to a DataGrid Web Server Control—`http://msdn.microsoft.com/en-us/vbcon/html/vbtskspecifyingpagingbehaviorindatagridwebcontrol.asp`

Sorting Data in a SQL Database—`http://msdn.microsoft.com/library/en-us/cpguide/html/cpconsortingdatainsqldatabase.asp`

13.12. Enabling Bi-Directional Sorting in a DataGrid

You want to allow users to click column headers in your DataGrid to sort that column, and to click the columns once again to reverse the order of the sort.

Technique

First you read an XML file into a dataset, and then bind the dataset to a DataGrid. Keep track of two attributes of the DataGrid—`SortExpression` and `SortDirection`. When a column is clicked, the `SortDataGrid` method is called. The grid is sorted by that column using `e.SortExpession`. If a column is clicked a second time, the sort direction is reversed.

The ASPX page:

```
      <asp:DataGrid id="SortGrid" runat="server"
CellPadding="5" AllowSorting="true" OnSortCommand="SortDataGrid">
        <AlternatingItemStyle BackColor="#ccffcc" />
        <HeaderStyle HorizontalAlign="center"
BackColor="#cccccc" Font-Bold="true" />
      </asp:DataGrid>
```

In `<script runat="server" />` block or codebehind:

```
      Protected Sub Page_Load([Source] As Object, e As EventArgs)
         If Not IsPostBack Then
            If SortGrid.Attributes("SortExpression") Is Nothing Then
               SortGrid.Attributes("SortExpression") = "HDate"
               SortGrid.Attributes("SortDirection") = "ASC"
            End If
            BindData()
         End If
      End Sub


      Private Sub BindData()
         Dim ds As New DataSet()
         Dim path As String = MapPath("BUD_WEN.xml")
         ds.ReadXml(path)
         Dim dv As DataView = ds.Tables(0).DefaultView

         Dim sortExpression As String = _
SortGrid.Attributes("SortExpression").ToString()
         Dim sortDirection As String = _
SortGrid.Attributes("SortDirection").ToString()
         dv.Sort = sortExpression + " " + sortDirection
```

```
        SortGrid.DataSource = dv
        SortGrid.DataBind()
    End Sub

    Public Sub SortDataGrid(sender As [Object], _
e As DataGridSortCommandEventArgs)
        Dim sortExpression As String = e.SortExpression
        Dim sortDirection As String = "ASC"
        If sortExpression.Equals(_
SortGrid.Attributes("SortExpression").ToString()) Then
            If SortGrid.Attributes("SortDirection").
➥ToString().StartsWith("ASC") Then
              sortDirection = "DESC"
            Else
              sortDirection = "ASC"
            End If
        End If
        SortGrid.Attributes("SortExpression") = sortExpression
        SortGrid.Attributes("SortDirection") = sortDirection

        BindData()
    End Sub
```

Comments

The DataGrid attributes should be set when the page first loads so that they do not have null values when you need to use them later. Set `SortExpression` to the default data field by which the grid is sorted.

The BindData method reads the XML data into a dataset. The sort expression and sort direction values are retrieved from the DataGrid attributes `SortExpression` and `SortDirection` and are used to sort the data. The data is then bound to the DataGrid.

The `SortDataGrid` method is called whenever one of the column headings is clicked. If the column that is clicked is the same as the value of the `SortExpression` attribute of the DataGrid, the sort direction is set to the opposite direction of `SortDirection`'s value.

See Also

Effective Sorting in ASP.Net DataGrids—
`http://msdn.microsoft.com/msdnnews/2001/sept/Sorting/Sorting.asp`

Sorting XML Data using the .NET DataGrid—
`http://www.codeproject.com/aspnet/XmlDataGrid.asp`

Bi-directional Sorting Without ViewState Enabled—`http://aspalliance.com/olson/articles/Bisort.aspx`

13.13. Editing Items in a DataGrid

You want to edit the contents of your DataGrid.

Technique

This example uses the Northwind database. Begin by inserting the DataGrid along with its attributes associated with making it editable.

Insert this as your HTML block:

```
<%@ Page Language="VB" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<script Runat="server" >

</script>
<html>
<head>
    <title>The Editable DataGrid</title>
</head>
<body>
    <h1>The Editable DataGrid
    </h1>
    <form runat="server">
        <asp:Label ID="ErrorLabel" ForeColor="#FF0000"
Font-Bold="True" Runat="server" />
        <asp:DataGrid id="myDataGrid" Runat="server"
AutoGenerateColumns="False" OnEditCommand="myDataGrid_Edit"
OnCancelCommand="myDataGrid_Cancel" OnUpdateCommand="myDataGrid_Update">
            <Columns>
                <asp:BoundColumn HeaderText="Customer ID"
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="CustomerID" ReadOnly="True" />
                <asp:BoundColumn HeaderText="Company"
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="CompanyName" />
                <asp:BoundColumn HeaderText="Name"
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="ContactName" />
```

```
                <asp:BoundColumn HeaderText="Title"
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="ContactTitle" />
                <asp:BoundColumn HeaderText="Phone No."
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="Phone" />
                <asp:BoundColumn HeaderText="Fax No."
                     HeaderStyle-Font-Size="12pt"
                     HeaderStyle-HorizontalAlign="Center"
                     HeaderStyle-BackColor="#FFCC00"
                     DataField="Fax" />
                <asp:EditCommandColumn ButtonType="LinkButton"
                     CancelText="Cancel" EditText="Edit"
                     UpdateText="Save" />
            </Columns>
        </asp:DataGrid>
    </form>
</body>
</html>
```

The following code will be inserted within your script block. If you plan to use this tool with codebehind, make sure that you make all of your DataGrid functions `Protected`. The reason for this is because you have a DataGrid control within your HTML block that is trying to access a function within your codebehind. Whenever you have a Web control within your HTML that is accessing any variable or function within your codebehind, that function or variable must be protected. For example, the `UpdateCommand` in this DataGrid would look like this if you were using codebehind:

```
Public ConnString As String = "SERVER=(local);
➥Database=Northwind;uid=user;pwd=password"

Sub BindData()
    '— Using the Try statement, we attempt to connect to our
    '— database, execute a SqlDataAdapter to store our data,
    '— populate a dataset and then bind that dataset
    '— to our DataGrid.
    Try
       Dim SqlConn As New SqlConnection(ConnString)
       Dim SqlString As String = & _
"SELECT CustomerID, CompanyName, ContactName, ContactTitle,
➥ Phone, Fax FROM Customers"
       Dim SqlComm As New SqlDataAdapter(SqlString, SqlConn)
       Dim customerData As New DataSet()
```

```
        SqlComm.Fill(customerData, "Customers")

        myDataGrid.DataSource = customerData
        myDataGrid.DataBind()

        SqlConn.Close()
        SqlComm.Dispose()
        SqlConn.Dispose()

    '— If we are not able to connect, display a friendly error
    Catch e As Exception
        ErrorLabel.Text = _
"Not able to connect to database. See description below: "
        ErrorLabel.Text += e.ToString()
    End Try
End Sub


Sub myDataGrid_Update(Sender As Object, e As DataGridCommandEventArgs)
    '— Take the data from each textbox in our editable item
    '— and assign that text to a string variable
    Dim CustomerID As String = Convert.ToString(e.Item.Cells(0).Text)
    Dim Company As String = CType(e.Item.Cells(1).Controls(0), TextBox).Text
    Dim Name As String = CType(e.Item.Cells(2).Controls(0), TextBox).Text
    Dim Title As String = CType(e.Item.Cells(3).Controls(0), TextBox).Text
    Dim Phone As String = CType(e.Item.Cells(4).Controls(0), TextBox).Text
    Dim Fax As String = CType(e.Item.Cells(5).Controls(0), TextBox).Text
    '— Again, using the Try statement, attempt to connect to our database
    '— and make an update with the data from our datagrid
    Dim SqlConn As New SqlConnection(ConnString)
    Try
        SqlConn.Open()
        Dim SqlString As String = "UPDATE Customers "
        SqlString &= "SET CompanyName = '" & _
Company.Replace("'", "''") & "', "
        SqlString &= "ContactName = '" + Name.Replace("'", "''") & "', "
        SqlString &= "ContactTitle = '" + Title.Replace("'", "''") & "', "
        SqlString &= "Phone = '" + Phone.Replace("'", "''") & "', "
        SqlString &= "Fax = '" + Fax.Replace("'", "''") & "'"
        SqlString &= " WHERE CustomerID = '" & CustomerID + "'"
        Dim SqlComm As New SqlCommand(SqlString, SqlConn)
        SqlComm.ExecuteNonQuery()
        SqlConn.Close()
        SqlComm.Dispose()
        SqlConn.Dispose()
    '— If for some reason we cannot connect, display a friendly error.
    Catch exc As Exception
```

```
        ErrorLabel.Text = _
"Not able to connect to database. See description below: "
        ErrorLabel.Text += exc.ToString()
    End Try
    '— Remove the edit focus
    myDataGrid.EditItemIndex = - 1
    '— Rebind our datagrid
    BindData()
End Sub


Sub myDataGrid_Cancel(Sender As Object, e As DataGridCommandEventArgs)
    '— Remove the edit focus
    myDataGrid.EditItemIndex = - 1
    '— Rebind our datagrid
    BindData()
End Sub


Sub myDataGrid_Edit(Sender As Object, e As DataGridCommandEventArgs)
    '— Set the edit focus to the item that was selected
    myDataGrid.EditItemIndex = CInt(e.Item.ItemIndex)
    '— Rebind our datagrid
    BindData()
End Sub


Sub Page_Load(Sender As Object, e As EventArgs)
    '— If this is not a postback event, bind the data to our datagrid
    If Not Page.IsPostBack Then
       BindData()
    End If
End Sub
```

Comments

Probably one of the simplest yet most efficient features of the DataGrid is its editing capability. Developers who have worked with classic ASP will probably remember how the editable DataGrid functionality worked. To code the functionality that the .NET DataGrid gives you today in classic ASP would take more than triple the time it takes for developers to create and present a fully functional editable .NET DataGrid.

See Also

http://aspalliance.com/Colt/Articles/Article3.aspx

http://aspalliance.com/andrewmooney/default.aspx?page=5

http://aspalliance.com/aldotnet/examples/dgautoscroll.aspx

13.14. Manipulating Individual Rows in a DataGrid

You want to manipulate or hide certain rows in a DataGrid.

Technique

A DataTable with four columns—Color, ColorName, Hexadecimal, and TurnedOn—is bound to a DataGrid. The `ItemDataBound` event looks at the ColorName column and makes the background of the Color column that color. The `ItemDataBound` event also looks at the TurnedOn column to determine whether the row is visible.

The ASPX page:

```
      <asp:DataGrid id="ColorGrid" runat="server" CellPadding="5"
OnItemDataBound="ColorGrid_ItemDataBound">
        <HeaderStyle HorizontalAlign="center" BackColor="#cccccc"
Font-Bold="true" />
      </asp:DataGrid>
      <br>
      <asp:Button id="RedButton" runat="server" Text="Red"
onclick="RedButton_Click" />
      <asp:Button id="GreenButton" runat="server" Text="Green"
onclick="GreenButton_Click" />
      <asp:Button id="BlueButton" runat="server" Text="Blue"
onclick="BlueButton_Click" />
      <asp:Button id="AllButton" runat="server" Text=" All "

onclick="AllButton_Click" />
```

In `<script runat="server" />` block or codebehind:

```
      Protected Sub Page_Load([Source] As Object, e As EventArgs)
         If Not IsPostBack Then
            BindData(True, True, True)
         End If
      End Sub

      Private  Sub BindData(ByVal RedOn As Boolean, _
ByVal GreenOn As Boolean, ByVal BlueOn As Boolean)
        Dim Colors As DataTable =  New DataTable()
        Colors.Columns.Add("Color", Type.GetType("System.String"))
        Colors.Columns.Add("ColorName", Type.GetType("System.String"))
        Colors.Columns.Add("Hexadecimal", Type.GetType("System.String"))
        Colors.Columns.Add("TurnedOn", Type.GetType("System.Boolean"))

        Dim Row1 As DataRow =  Colors.NewRow()
        Row1("Color") = " "
```

```
        Row1("ColorName") = "Red"
        Row1("Hexadecimal") = "#ff0000"
        Row1("TurnedOn") = RedOn
        Colors.Rows.Add(Row1)

        Dim Row2 As DataRow =  Colors.NewRow()
        Row2("Color") = " "
        Row2("ColorName") = "Green"
        Row2("Hexadecimal") = "#00ff00"
        Row2("TurnedOn") = GreenOn
        Colors.Rows.Add(Row2)

        Dim Row3 As DataRow =  Colors.NewRow()
        Row3("Color") = " "
        Row3("ColorName") = "Blue"
        Row3("Hexadecimal") = "#0000ff"
        Row3("TurnedOn") = BlueOn
        Colors.Rows.Add(Row3)

        ColorGrid.DataSource = Colors
        ColorGrid.DataBind()
      End Sub


      Protected  Sub ColorGrid_ItemDataBound(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.DataGridItemEventArgs)
        If e.Item.ItemType = ListItemType.Item Or e.Item.ItemType =
ListItemType.AlternatingItem Then
          e.Item.Cells(0).BackColor = Color.FromName(e.Item.Cells(1).Text)
          e.Item.Visible = Convert.ToBoolean(e.Item.Cells(3).Text)
        End If
      End Sub

      Protected  Sub RedButton_Click(ByVal Source As Object,
        ByVal e As EventArgs)
          BindData(True, False, False)
      End Sub

      Protected  Sub GreenButton_Click(ByVal Source As Object,
        ByVal e As EventArgs)
          BindData(False, True, False)
      End Sub

      Protected  Sub BlueButton_Click(ByVal Source As Object,
        ByVal e As EventArgs)
          BindData(False, False, True)
      End Sub
```

```
Protected  Sub AllButton_Click(ByVal Source As Object,
  ByVal e As EventArgs)
    BindData(True, True, True)
End Sub
```

Comments

As the DataGrid binds the data from the DataTable, the `ItemDataBound` event is fired for each row of the DataTable. If the row in the `ItemDataBound` event is a `ListItemType` or an `AlternatingItem` type, two things happen:

- The `BackColor` of the Color column changes to the color indicated in the ColorName column.
- The TurnedOn column shows or hides the row depending on whether the property `visible` is set to `true` or `false`.

Note that if your DataGrid allows the users to select certain rows, you should also check for the `SelectedItemType`.

The `Click` event for each of the buttons calls the BindData method with the parameters telling which rows to show or hide. Changing the content of the DataTable causes the DataGrid to appear different by manipulating the rows in the `ItemDataBound` event.

See Also

Section 13.7, "Data-binding to a DataGrid"

ItemDataBound Event—`http://msdn.microsoft.com/library/en-us/cpref/html/frlrfsystemwebuiwebcontrolsdatagridclassitemdataboundtopic.asp`

13.15. Implementing a Master-Detail Report Using Two DataGrids

You want to create a master-detail view of some data using two DataGrid controls.

Technique

There are two master and details DataGrid controls, in which the master DataGrid lists the information of authors, whereas the details DataGrid lists the information of titles of the author. When you select an author from the master DataGrid, the information corresponding to the selected author is displayed in the details DataGrid.

The ASPX page:

```
        <asp:datagrid id="MasterDataGrid" runat="server"
OnPageIndexChanged="MasterDataGrid_PageIndexChanged" PageSize="6"
```

```
AllowPaging="True" OnSelectedIndexChanged="MasterDataGrid_SelectedIndexChanged"
 DataKeyField="Last Name" Width="100%">
                <Columns>
                    <asp:ButtonColumn Text="Show details"
CommandName="Select" />
                </Columns>
            </asp:datagrid>
            <br />
            <asp:datagrid id="DetailsDataGrid" runat="server"
EnableViewState="False" Width="100%"/>
        </p>


        <p/>
            <asp:Label id="ErrorLabel" runat="server"/>
```

In `<script runat="server" />` block or codebehind:

Both of the master and details DataGrid must be databound during `Page_Load`, as demonstrated here:

```
    Sub Page_Load(Sender As Object, E As EventArgs)

        If Not IsPostBack Then

            MasterDataGrid.SelectedIndex = 0
            BindMasterDataGrid()
            BindDetailDataGrid()

        End If

    End Sub
```

When you click the Show Details Button on the master DataGrid, its `SelectedIndexChanged` event handler will be fired:

```
    Protected Sub MasterDataGrid_SelectedIndexChanged(Sender As Object, E As
EventArgs)
        BindDetailDataGrid()
    End Sub
```

The `SelectedIndexChanged` event handler will then fire the data-binding of the details DataGrid, based on the primary key of the selected grid item row. That primary key is retrieved from the DataKeys collection of the master DataGrid.

```
    Sub BindDetailDataGrid()

        If MasterDataGrid.SelectedIndex <> -1 Then

            Dim ConnectionString As String = _
```

```
"server=Localhost;database=pubs;trusted_connection=true"


            Dim FilterValue As String
            Try
               FilterValue = CStr(MasterDataGrid.DataKeys( _
                             MasterDataGrid.SelectedIndex)).Replace("'", "''")
            Catch
               FilterValue = ""
            End Try


            Dim CommandText As String = "select title as Title, price as Price,
➥ ytd_sales as [YTD Sales] from titleview where au_lname = '"
➥ & FilterValue & "'"

            Try
                Dim objConnection As New SqlConnection(ConnectionString)
                Dim objCommand As New SqlCommand(CommandText, objConnection)

                objConnection.Open()

                DetailsDataGrid.DataSource = _
                   objCommand.ExecuteReader(CommandBehavior.CloseConnection)
                DetailsDataGrid.DataBind()

            Catch SqlEx As SqlException
                ErrorLabel.Text = SqlEx.Message
            Catch Ex As Exception
                ErrorLabel.Text = Ex.Message
            Finally
                objConnection.Close()


            End Try

        End If

    End Sub
```

After passing the primary key to the `BindDetailDataGrid` procedure, the example binds and displays the author's title information on the detail DataGrid.

Comments

Using two DataGrids to display data with a master-detail relationship is very useful, especially when you're showing some nested or hierarchical data, such as menu and submenus.

13.16. Rendering Images in a DataGrid

You want to display an image in each row of a DataGrid.

Technique

You can add images to a DataGrid by using a template column. This example uses the template column to display the photos of the top five authors of this book. You start off by creating a new DataGrid and adding two columns to it. You then set the first column to `TemplateColumn` and begin to set up the `ItemTemplate`. You create the first half of the HTML `<img>` tag. When setting the source attribute, be sure to use single quotes to encapsulate the image URL so that you can use double quotes within the data-binding syntax.

Add the following code to data-bind the Photo column to the `source` attribute:

```
<%#DataBinder.Eval(Container.DataItem, "Photo")%>
```

Then you close the `<img>` tag. Finally, you set the second column to a `BoundColumn` and bind it to the Name column.

The ASPX page is as follows:

```
<%@Import namespace="System.Data"%>
<HTML>
   <body>
      <form id="form1" method="post" runat="server">
         <asp:DataGrid ID="Authors" Runat="server" AutoGenerateColumns="False">
            <Columns>
               <asp:TemplateColumn HeaderText="Picture">
                  <ItemTemplate>
                     <img
                      src='<%#DataBinder.Eval(Container.DataItem, "Photo")%>'>
                  </ItemTemplate>
               </asp:TemplateColumn>
               <asp:BoundColumn DataField="Name" HeaderText="Name" />
            </Columns>
         </asp:DataGrid>
      </form>
   </body>
</HTML>
```

The technique for populating a DataTable and data-binding it isn't relevant to this example, so it's not covered in a lot of detail. In short, you create a new DataTable and add a photo and a name column to it. Then, you call the `AddRow` function to add the authors to the table and data-bind the DataTable to the DataGrid.

In `<script runat="server" />` block or codebehind:

```
Sub Page_Load(sender As Object, e As EventArgs)
   Dim authorsTable As DataTable
   authorsTable=new DataTable()
   authorsTable.Columns.Add("Photo")
   authorsTable.Columns.Add("Name")
   AddRow(authorsTable, "recipe1316a.jpg", "Steven Smith")
   AddRow(authorsTable, "recipe1316b.jpg", "Jeremy Zongker")
   AddRow(authorsTable, "recipe1316c.jpg", "Haroon Malik")
   AddRow(authorsTable, "recipe1316d.jpg", "Rob Howard")
   AddRow(authorsTable, "recipe1316e.jpg", "Daniel Olson")
   Authors.DataSource=authorsTable
   Authors.DataBind()
End Sub

Sub AddRow(AuthorsTable As DataTable, Photo As String, Name As String)
   Dim row As DataRow
   row=AuthorsTable.NewRow()
   row("Photo")=Photo
   row("Name")=Name
   AuthorsTable.Rows.Add(row)
End Sub
```

See Also

Section 10.10, "Displaying an Image from SQL Server"

Section 13.7, "Data-binding to a DataGrid"

Section 13.8, "Using Different Column Types in a DataGrid"

13.17. Adding a Confirmation Box to a DataGrid Button

You want to add a confirmation dialog box to a button in a DataGrid.

Technique

Consider a Delete button in a DataGrid. Whenever you want to delete a record in the DataGrid, you want to trigger an `onclick` event and ask for your confirmation before actual deletion. This would avoid accidental data loss. You can use two approaches to display a button in a DataGrid for deletion—using the built-in Button column or using a Template column with a Button control.

The ASPX page is as follows:

```
<asp:datagrid id="myDataGrid" runat="server"
   OnItemDataBound="DataGrid_ItemDataBound"
   DataKeyField="au_id"
   OnDeleteCommand="DataGrid_Delete"
   AllowPaging="True"
   PageSize="6"
   OnPageIndexChanged="DataGrid_Page"
   AutoGenerateColumns="False">

   <PagerStyle mode="NumericPages"/>
   <Columns>
      <asp:ButtonColumn Text="Delete" CommandName="Delete"/>
      <asp:BoundColumn DataField="au_id" HeaderText="Author ID"/>
      <asp:BoundColumn DataField="au_fname" HeaderText="Last Name"/>
      <asp:BoundColumn DataField="au_lname" HeaderText="First Name"/>
      <asp:TemplateColumn>
         <ItemTemplate>
            <asp:Button id="DeleteButton" Runat="Server"
               Text="Delete"
               OnCommand="DeleteButton_Click"
               CommandArgument='<%# Container.ItemIndex %>'/>
         </ItemTemplate>
      </asp:TemplateColumn>
   </Columns>
</asp:datagrid>
```

You can locate the button by finding and casting it for each item of the DataGrid, and then add a JavaScript method to the attributes collection of the button in the `ItemDataBound` event of the DataGrid.

In `<script runat="server" />` block or codebehind:

```
Sub DataGrid_ItemDataBound(Sender As Object, e As DataGridItemEventArgs)

    If e.Item.ItemType = ListItemType.Item Or _
       e.Item.ItemType = ListItemType.AlternatingItem Then

        Dim LinkButtonControl As LinkButton = _
CType(e.Item.Cells(0).Controls(0), LinkButton)
        LinkButtonControl.Attributes.Add("onclick", _
"Javascript:return confirm('Are you sure?');")

        Dim ButtonControl As Button = _
CType(e.Item.FindControl("DeleteButton"), Button)
        ButtonControl.Attributes.Add("onclick", _
"Javascript:return confirm('Are you sure?');")
    End If
End Sub
```

Comments

When an item is data-bound to the DataGrid control, you can add a JavaScript method to ask for confirmation. This will display a confirmation dialog box, which means the click is processed and the event is executed only when the users confirm that they are sure they want to do so.

In the `ItemDataBound` event, you can explicitly locate the Button control from its cell and control index, or you can use the `FindControl` method to do this. However, no matter which approach you use to locate the Button control, you have to cast it into the correct type so that the `OnClick` client script can be added to it successfully.

See Also

Section 13.18, "Implementing Support for Double-Clicks in a DataGrid"

13.18. Implementing Support for Double-Clicks in a DataGrid

You want to select an item by double-clicking anywhere in the DataGrid row.

Technique

Start with an existing DataGrid that is populated from a data source (see example 13.7), or download the full example for this recipe from the book's Web site. In the `ItemDataBound` event handler of DataGrid, you assign a JavaScript `ondblclick` method to the DataGrid item.

The ASPX page is as follows:

```
        <asp:DataGrid id="DblClickDataGrid"
OnItemDataBound="DblClickDataGrid_ItemDataBound" Runat="Server">
            <SelectedItemStyle backcolor="Yellow" />
            <Columns>
                <asp:ButtonColumn Text="Select" HeaderText="Select"
CommandName="Select"/>
            </Columns>
        </asp:DataGrid>
```

You add a JavaScript method to the DataGrid item in the `ItemDataBound` event handler of the DataGrid.

In `<script runat="server" />` block or codebehind:

```
Sub DblClickDataGrid_ItemDataBound(Sender As Object, _
e As DataGridItemEventArgs)
    If e.Item.ItemType = ListItemType.Item Or e.Item.ItemType = _
ListItemType.AlternatingItem Then
        e.Item.Attributes.Add("ondblclick", _
```

```
"Javascript:__doPostBack('myDblClick','" & e.Item.ItemIndex & "');")
    End If
End Sub
```

If the page is being posted back, check and parse the index from `__EventTarget`, and then set the corresponding `SelectedIndex` of DataGrid.

```
Sub Page_Load(Sender As Object, E As EventArgs)
    If Not IsPostBack Then
        BindGrid()
    Else
        If Not Request.Form("__EventTarget") Is Nothing And _
           Request.Form("__EventTarget") = "myDblClick" Then
            DblClickDataGrid.SelectedIndex = _
CInt(Request.Form("__EVENTARGUMENT"))
        End If
    End If
End Sub
```

Comments

In addition to the classic Select button of a DataGrid, users can select an item by double-clicking any object on the DataGrid. A client-side double-click method was added to the attributes collection of the `DataGridItem`. Because this page uses a postback event, ASP.NET generates the `__doPostBack` JavaScript method to the page automatically, which consists merely of a form submission.

A DataGrid is basically an HTML table, whereby each cell per DataGridItem is equivalent to a `<td>` and each grid row is equivalent to a `<tr>` element. By assigning the JavaScript `ondblclick` method to the table row, you can select an item by double-clicking anywhere on a grid item row. This is similar to selecting a row by clicking the Select button column in a DataGrid.

> **See Also**
> Selecting Rows by Clicking Anywhere—`http://msdn.microsoft.com/library/en-us/dv_vstechart/html/vbtchTopQuestionsAboutASPNETDataGridServerControl.asp`

13.19. Implementing a Check Box Column in a DataGrid

You want to implement a check box column in a DataGrid.

Technique

If you add a Template column with a CheckBox control to a DataGrid, your users can select multiple items in a DataGrid for a bulk operation.

The ASPX page is as follows:

```
<asp:datagrid id="myDataGrid" runat="server" AutoGenerateColumns="False">
   <Columns>
      <asp:TemplateColumn>
         <HeaderTemplate>
            <input type="CheckBox" name="SelectAllCheckBox"
onclick="SelectAll(this)">
         </HeaderTemplate>
         <ItemTemplate>
            <asp:CheckBox id="SelectCheckBox" Runat="Server" />
         </ItemTemplate>
      </asp:TemplateColumn>
      <asp:BoundColumn DataField="pub_id" HeaderText="ID"/>
      <asp:BoundColumn DataField="pub_name" HeaderText="Name"/>
      <asp:BoundColumn DataField="city" HeaderText="City"/>
      <asp:BoundColumn DataField="state" HeaderText="State"/>
      <asp:BoundColumn DataField="country" HeaderText="Country"/>
   </Columns>
</asp:datagrid>
```

Using the `TemplateColumn` with the CheckBox control, you can select multiple items in the DataGrid, or click the check box in the `HeaderTemplate` of this CheckBox column to (de)select all items in one click.

After selecting the items, you can click a button and then walk through the items in the DataGrid. You can find and cast the CheckBox control from the appropriate column to determine whether the check box is checked.

In `<script runat="server" />` block or codebehind:

```
Function GetSelectedPublishers() As String
   Dim i As Integer

   For i = 0 To myDataGrid.Items.Count - 1
      If CType(myDataGrid.Items(i).FindControl("SelectCheckBox"), _
CheckBox).Checked Then
         If PublisherIDs <> "" Then PublisherIDs &= ","
         PublisherIDs &= myDataGrid.DataKeys(i)
      End If
   Next
   GetSelectedPublishers = PublisherIDs
End Function
```

In the previous function, the items in the DataGrid are looped through and the CheckBox control is found by using `FindControl` method. Therefore, the `DataKeyField` of the current `DataGridItem`, which is normally the primary key of the working table, is distinguished and appended to form a collection. This concatenated string will be used for the later bulk operation (such as a delete or copy opration).

For example, using the `GetSelectedPublishers()` method, you could implement a delete event as follows:

```
Sub DeleteButton_Click(sender As Object, e As EventArgs)
    Dim PublisherIDs As String = GetSelectedPublishers()
    Dim ConnectionString As String = _
"server=(local);database=pubs;uid=user;pwd=password"
    Dim CommandText As String = _
"delete from Publishers where pub_id in (" & PublisherIDs & ")"
    Dim objConnection As New SqlConnection(ConnectionString)
    Dim objCommand As New SqlCommand(CommandText, objConnection)

    Try
        objConnection.Open()
        objCommand.ExecuteNonQuery()
    Catch SqlEx As SqlException
        MessageLabel.Text = SqlEx.Message & "<br>" & CommandText
    Catch Ex As Exception
        MessageLabel.Text = Ex.Message
    Finally
        objConnection.Close()
    End Try
    BindGrid()
End Sub
```

The `BindGrid()` method:

```
Sub BindGrid()
    Dim ConnectionString As String = _
"server=(local);database=pubs;uid=user;pwd=password"
    Dim CommandText As String = "select * from Publishers"
    Dim objConnection As New SqlConnection(ConnectionString)
    Dim objCommand As New SqlCommand(CommandText, objConnection)

    Try
        objConnection.Open()
        myDataGrid.DataSource = _
objCommand.ExecuteReader(CommandBehavior.CloseConnection)
        myDataGrid.DataBind()
    Catch SqlEx As SqlException
        MessageLabel.Text = SqlEx.Message
    Catch Ex As Exception
        MessageLabel.Text = Ex.Message
        Finally
          objConnection.Close()
    End Try
End Sub
```

Comments

You can use the `FindControl` method to locate the CheckBox control in the `TemplateColumn` of DataGrid, whereas you can also use the standard approach—getting a control by specifying its column and control index. If you are getting a CheckBox control via its control index, bear in mind that the CheckBox control is usually the second control in the `TemplateColumn` (index 1) because a literal control precedes it.

In the `HeaderTemplate` of the CheckBox column, you can put a check box HTML control and fire a simple client-side method to fill in the check boxes all at once.

See Also

Andy Smith's RowSelectorColumn control—
`http://www.metabuilders.com/Tools/RowSelectorColumn.aspx`

13.20. Implementing a DropDownList in Edit Mode in a DataGrid

You want to use a DropDownList for one of the fields in your DataGrid when in Edit mode, and then retrieve the value of this DropDownList when the user saves the item.

Technique

To provide a DropDownList when a user edits a record, you can put it in the `EditItemTemplate` of a TemplateColumn in a DataGrid.

The ASPX page is as follows:

```
<asp:TemplateColumn HeaderText="Region">
   <ItemTemplate>
      <asp:Label id="regionDescriptionLabel" runat="server"
    Text='<%# DataBinder.Eval(Container.DataItem, "RegionDescription") %>'/>
   </ItemTemplate>
   <EditItemTemplate>
      <asp:DropDownList id="regionDropDownList" Runat="Server"
         DataSource="<%# GetRegionDataTable() %>"
         DataTextField="RegionDescription"
         DataValueField="RegionID" />
   </EditItemTemplate>
</asp:TemplateColumn>
```

After adding a DropDownList to the EditItemTemplate of a DataGrid, you have to bind some data to this control, and set its display text and corresponding values. For the data-binding process, you can retrieve the data from a database and construct a DataTable, and

then you can return this DataTable and bind it to the DropDownList. This data-binding takes place in the `ItemDataBound` event of the DataGrid, shown here:

```
Sub myDataGrid_ItemDataBound(ByVal sender As Object, _
ByVal e As DataGridItemEventArgs)
    If e.Item.ItemType = ListItemType.EditItem Then
        Dim objDataRowView As DataRowView = CType(e.Item.DataItem, DataRowView)
        Dim currentgenre As String = CType(objDataRowView(2), String)
        Dim ctlDropDownList As DropDownList = _
CType(e.Item.FindControl("regionDropDownList"), DropDownList)
        ctlDropDownList.SelectedIndex =
ctlDropDownList.Items.IndexOf(ctlDropDownList.items.findbytext(currentgenre))
    End If
End Sub
```

Comments

In `EditItemTemplate` of the DataGrid, the datasource of the DropDownList is set to a function, which returns a DataTable. In this function, the data can be retrieved from the database once, and then the DataTable can be inserted into the cache to reduce the number of database hits required. The complete source for this is available online.

> **See Also**
> DataGrid In-place Editing—`http://msdn.microsoft.com/msdnmag/issues/01/06/cutting/default.aspx`

13.21. Editing Items in a DataList

You want to edit an item from within a DataList.

Technique

Import the `System.Data` and `System.Data.SqlClient` namespaces for this example.

The ASPX page is as follows:

```
<asp:DataList id="DL1" runat="server" OnCancelCommand="CancelItem"
OnDeleteCommand="DeleteItem" OnUpdateCommand="UpdateItem"
OnEditCommand="EditItem" DataKeyField="au_id" EditItemStyle-BackColor="yellow"
CellSpacing="2">
   <HeaderTemplate>
      <h1>List of Authors
      </h1>
   </HeaderTemplate>
   <ItemTemplate>
```

```
      <%# Container.DataItem("au_fname") %>  
<%# Container.DataItem("au_lname") %>
      <br />
      <%# Container.DataItem("Address") %>
      <br />
      <asp:Button CommandName="Edit" Text="Edit Record" runat="server" />
   </ItemTemplate>
   <SeparatorTemplate>
      <hr width="20%" />
   </SeparatorTemplate>
   <AlternatingItemStyle backcolor="Lime"></AlternatingItemStyle>
   <EditItemTemplate>
      <b>First Name</b>
      <asp:TextBox id="FNameTextBox"
Text='<%# Container.DataItem("au_fname") %>' runat="server" />
      <br />
      <b>Last Name</b>
      <asp:TextBox id="LNameTextBox"
Text='<%# Container.DataItem("au_lname") %>' runat="server" />
      <br />
      <b>Address</b>
      <asp:TextBox id="AddressTextBox"
Text='<%# Container.DataItem("Address") %>' runat="server" />
      <br />
      <asp:Button CommandName="Update" Text="Update" runat="server" />
       
      <asp:Button CommandName="Delete" Text="Delete" runat="server" />
       
      <asp:Button CommandName="Cancel" Text="Cancel" runat="server" />
   </EditItemTemplate>
   <FooterTemplate>
      <hr width="100%" />
   </FooterTemplate>
</asp:DataList>
```

In order to use a DataList for editing, certain properties must be set. The `EditItemTemplate` setting specifies the appearance of the list when the selected record in the DataList switches it to editing mode. Other attributes of the DataList control, such as the `OnEditCommand`, `OnUpdateCommand`, `OnDeleteCommand`, and `OnCancelCommand` controls, are passed the name of the events that are to be raised. These events are defined within the `<script>` blocks or the codebehind file.

The second important property is the `DataKeyField`, which should be set to the name of the primary key column in the datasource.

In `<script runat="server" />` block or codebehind:

```
Private Sub Page_Load(Source As Object, E As EventArgs)
      If Not Page.IsPostback Then
```

```
            BindToList()
        End If
        'Clear any previous messages...
        MessageLabel.Text = ""
End Sub

Public Sub BindToList()
        Dim _Connection As New SqlConnection( _
"Server=localhost; Database=pubs; uid=user; pwd=password;")
        Dim _Command As New SqlCommand( _
"Select au_id, au_fname, au_lname, address From authors", _Connection)

        Try
            _Connection.Open()
            DL1.DataSource = _Command.ExecuteReader
            DL1.DataBind()
        Catch _Error As Exception
            MessageLabel.Text = _Error.Message
        Finally
            _Connection.Close()
        End Try
End Sub

Private Sub EditItem(Source As Object, E As DataListCommandEventArgs)
        'Set EditItemIndex property to the index of the record raising the event
        DL1.EditItemIndex = E.Item.ItemIndex
        BindToList()
End Sub

Private Sub UpdateItem(Source As Object, E As DataListCommandEventArgs)
        'Get References to the Textboxes...
        Dim _FName, _LName, _Address As TextBox

        _FName = CType(E.Item.FindControl("FNameTextBox"), TextBox)
        _LName = CType(E.Item.FindControl("LNameTextBox"), TextBox)
        _Address = CType(E.Item.FindControl("AddressTextBox"), TextBox)

         'Create the query...
        Dim _Query As String
        _Query = "Update authors Set au_fname='" & _FName.Text & _
"', au_lname='" & _LName.Text & "', address='" & _Address.Text & _
"' Where au_id='" & DL1.DataKeys(E.Item.ItemIndex) & "'"

        'Update the data source...
        Dim _Connection As New SqlConnection( _
"Server=localhost; Database=pubs; uid=user; pwd=password;")
        Dim _Command As New SqlCommand(_Query, _Connection)
```

```
      Try
          _Connection.Open()
          _Command.ExecuteNonQuery
              'Switch off the Edit mode
          DL1.EditItemIndex = -1
          BindToList()
      Catch _Error As Exception
          MessageLabel.Text = _Error.Message
      Finally
          _Connection.Close()
      End Try
End Sub

Private sub DeleteItem(Source As Object, E As DataListCommandEventArgs)
      'Create the query...
      Dim _Query As String
      _Query = "Delete From authors Where au_id='" & _
DL1.DataKeys(E.Item.ItemIndex) & "'"

      'Update the data source...
      Dim _Connection As New SqlConnection( _
"Server=localhost; Database=pubs; uid=user; pwd=password;")
      Dim _Command As New SqlCommand(_Query, _Connection)

      Try
          _Connection.Open()
          _Command.ExecuteNonQuery
          'Switch off the Edit mode
          DL1.EditItemIndex = -1
          BindToList()
      Catch _Error As Exception
          MessageLabel.Text = _Error.Message
      Finally
          _Connection.Close()
      End Try
End Sub

Private Sub CancelItem(Source As Object, E As DataListCommandEventArgs)
      'Switch off the Edit mode...
      DL1.EditItemIndex = -1
      BindToList()
End Sub
```

Comments

All the events and methods defined in this section are self-explanatory. The `EditItem()` method is responsible for enabling the editing mode of the DataList control. The `Update` and `Delete` event methods update or delete the currently selected record, whose index is stored in the `Index` property of the `DataListEventArgs` object.

The `UpdateItem()`, `DeleteItem()`, and `CancelItem()` events, when raised, also disable the editing mode of the DataList control and refresh it by calling the `BindToList()` method. The `BindToList()` method queries the datasource for records and then displays them in the control.

See Also

Section 13.4, "Data-binding to a DataList"

Section 13.5, "Implementing Sorting in a DataList"

Section 13.6, "Implementing Paging in a DataList"